Working Paper

Document de travail

Working Group on International Macroeconomics

Groupe de travail en macroéconomie internationale

Department of Finance

Ministère des Finances

101-06198/

INTERMOD Model User's and Developer's Guide for PC Systems

by

P. Bagnoli and A. Kruus

Working Paper No. 88-9

The Working Group on International Macroeconomics
Department of Finance, Ottawa

This paper comprises a two-part manual for use with the PC version of INTERMOD. Part I represents work performed jointly by Philip Bagnoli and Anders Kruus while Part II represents work performed by Philip Bagnoli. The authors wish to thank the members of the Working Group on International Macroeconomics for comments and suggestions and most especially John Helliwell for extensive comments regarding the contents of this paper. The authors retain sole responsibility for any remaining errors.

Working papers are intended to make analytic work undertaken in the Department of Finance available to a wider readership. They have received only limited evaluation and do not necessarily reflect the views of the Department of Finance. Comments on the working papers are invited and may be sent to the author(s). They are circulated in the language of preparation.

Single copies of the SIMULATOR diskette referred to in the "INTERMOD Model User's and Developer's Guide for PC Systems" working paper may be obtained by writing to:

Distribution Center
Department of Finance
P1 West Tower
L'Esplanade Laurier
140 O'Connor Street
Ottawa, Ontario K1A 0G5
Canada

(or telephoning (613) 995-2855)

and requesting the "INTERMOD SIMULATOR diskette". Please indicate whether a 5 1/4" or 3 1/2" diskette is required.

ABSTRACT

This paper presents a guide to utilising the PC version of the INTERMOD model. The first part of the guide presents instructions intended primarily for the user interested in probing model properties. They are presented in a manner that assumes the user has little experience with IBM PCs but possesses a basic understanding of macroeconomic modelling. The PC version of INTERMOD makes use of a series of screens and a help facility to permit model simulation to be performed easily.

The second part of the guide is intended to assist the user whose primary interest is to modify or develop models for use with the PC simulator so that alternative theoretical structures for INTERMOD can be tested. These instructions are presented at a more technical level than the basic usage instructions. The user is assumed to have a minimal understanding of computer programming since model equations must be coded (in Fortran) and compiled during the model building process. However, the information provided in the body and appendices of this paper should be sufficient to preclude the need for specific training in Fortran programming.

RÉSUMÉ

Ce document constitue un mode d'emploi de la version PC (ordinateur personnel) du modèle INTERMOD. Dans la façon de présenter les instructions, on suppose que l'usager sait déjà utiliser un PC d'IBM et qu'il connaît les principes de base de l'établissement de modèles macroéconomiques. La version PC du modèle INTERMOD fait appel à une série d'images-écrans, doublée d'une fonction d'aide, et permet ainsi d'effectuer très rapidement une analyse macroéconomique avec le modèle de base.

Ce document présente aussi les instructions nécessaires pour modifier ou élaborer des modèles à utiliser avec un simulateur PC de façon à pouvoir mettre à l'essai divers fondements théoriques du modèle INTERMOD. Bien entendu, ces instructions sont libellées dans un langage plus technique que les instructions de base. On présume que l'usager connaît les rudiments de la programmation d'un ordinateur, car les équations doivent être codées et compilées pendant la construction du modèle. Ces équations doivent être introduites en langage FORTRAN, mais il n'est pas nécessaire d'avoir reçu une formation pour pouvoir utiliser ce langage, car les renseignements donnés à cette fin dans le corps et dans les annexes de ce document sont suffisamment clairs.

TABLE OF CONTENTS

PART I - Model User's Guide

I Introduction 1

II System Requirements and Set-Up 2

- Co-processors only
- Check the CONFIG.SYS file
- Adaptive versus Consistent Expectations
- Installing SIMULATOR on a hard disk

III A Quick Hands-on Introduction 4

IV Overview of Simulator 5

- Help facility built into SIMULATOR
- SIMULATOR's structure
- Brief description of some files

V Example #1: A US Money Supply Shock 8

- Starting SIMULATOR
- Update default files list
- Set simulation start and duration
- Update display title
- Set up run characteristics
- Set simulation parameters
- Save changes in start-up file
- Read a start-up file
- Simulate
- Data routines

VI Examining Results 17

- SHOW
- PRINT

VII Example #2: An Oil Price Shock 19

- Creating a new control solution
- Resetting coefficients
- Using a consistent expectations structure

VIII Example #3: Changing Exchange Rates 24

- Revising residuals

PART II - Model Builder's Guide

I Introduction 25

II Model Building 26

(1) Structure of model sector coding
- putting together a Fortran subroutine

(2) The ORDER.FOR program
- setting lags and leads
- connecting subroutines to simulator

(3) Putting the model together
- linking all programs together

III Data Structures 29

- how the database is linked to the simulator

(1) Technical note
- describes Fortran data structures

IV Model Building Utilities 31

(1) Build file to link data to variables
- creating an EQNAME.FOR file

(2) Build a binary database
- convert an ASCII database to a binary database

(3) Create model to generate residuals
- replace dependent variables in model

(4) Generate residuals
- use model to build a database with the residuals

V Summary 34

- brief outline of flow of model building
- checklist of steps to build model

VI Revision of Existing Models 34

- reduced set of instructions to revise an existing model

VII Miscellaneous Tips 38

- helpful hints to finding problems in model

Appendices I, II, III and IV

PART I - Model User's Guide

I Introduction

SIMULATOR[1] is a FORTRAN-based software package that allows fast and simple simulations of macroeconomic models on a PC. The particular model coded into SIMULATOR is the INTERMOD international macroeconomic model, developed at the Department of Finance. Other models can be designed and implemented within the SIMULATOR framework. However, for the purpose of exposition, this manual deals exclusively with the INTERMOD model. Users wishing to implement their own model or modify INTERMOD should refer to the Simulator Model Builder's Manual.

INTERMOD and SIMULATOR are presently used for policy analysis, rather than forecasting; their primary purpose is to predict the impacts of particular macroeconomic policies or events. If the model is to be used for forecasting you should carefully examine the exogenous variables of the model along with the residuals of the stochastic equations to satisfy yourself that these are reasonable settings.

HELP options and default settings are provided in the program to allow you to start full use of SIMULATOR as soon as possible. We suggest that you initially invest the time required to run a few basic exercises using the default settings, thereby building up your familiarity and ease with SIMULATOR. This will prove helpful when attempting to expand the scope of the policy simulations.

The SIMULATOR software is not a commercial package and is distributed free of charge or obligation; we ask only that you assist us in improving SIMULATOR by communicating to us suggestions or comments you have regarding the software package or the macroeconomic model incorporated within it.

---

[1]The software comprising the simulator program is based on mainframe simulation programs developed in the late 1960s and early 1970s for the RDX series of models at the Bank of Canada. Further refinements to the mainframe programs were conducted at the University of British Columbia by Robert McRae with funding from the Social Sciences and Humanities Research Council (SSHRCC). Philip Bagnoli started the PC developments while at the University of British Columbia, also with SSHRCC support. Final development into its present form was conducted at the Department of Finance, Ottawa by Philip Bagnoli.

II System Requirements and Set-Up

Co-processors only

SIMULATOR can be run on any IBM or compatible personal computer, provided the system has a math co-processor chip installed in it. The math co-processor is an option on PCs that must be ordered separately. If you did not specifically request a co-processor when you purchased your system then you most likely do not have one. The most straightforward manner to determine whether your system has one if you are uncertain is to simply run SIMULATOR. It will return an error message if a co-processor is not detected.

Check the CONFIG.SYS

Before SIMULATOR can be run, your PC system must be a suitably configured. In practical terms, this means that your basic boot-up files should include a CONFIG.SYS file. To be able to operate SIMULATOR, this CONFIG.SYS file must include the command lines:

```
files   = 20
buffers = 10
```

If your CONFIG.SYS file does not have these commands, or if you do not have a CONFIG.SYS file at all, you should consult your DOS manual for instructions regarding the creation or editing of a CONFIG.SYS file. You must reboot the system with a proper CONFIG.SYS file in place in your root directory before you start SIMULATOR. If your setting for the parameters is already greater than these settings then there will be no need to modify the file.

Adaptive versus Consistent Expectations

SIMULATOR is available in two forms; one designed to run INTERMOD with an adaptive ("backward-looking") expectations structure, and one for running the model-consistent ("rational") expectations version of INTERMOD. The simulation run time using the consistent expectations version of SIMULATOR can be over 100 times that of a simulation using adaptive expectations (depending on the nature of the shock). Both versions of the model are contained in the program SIM.EXE. As described later, you can switch between the two expectations schemes by selecting an option in one of the menu items. The differences between the two models are found in the term structure of interest rates for each country, the calculation of developing countries' debt, the exchange rate

expectations and absorption price deflator expectations. By selecting a particular expectations scheme you will be selecting which equations will be used in the model.

Installing SIMULATOR on your PC's hard disk

To install SIMULATOR onto your hard disk, simply follow these instructions:

1) type **MD C:\SIMULATE** - this creates a sub-directory for SIMULATOR

2) type **CD C:\SIMULATE** - this sets the SIMULATE directory as the default directory

3) insert the SIMULATOR diskettes into your a: [or b:] disk drive

4) type **COPY A:*.*** [or **B:*.***] **C:** - this copies over the files on the diskettes to your hard disk drive

5) Simulator is now installed on your hard disk

III A Quick Hands-On Introduction

This section presents a condensed run through SIMULATOR intended to demonstrate a sample simulation exercise. For details concerning the various steps presented here please refer to other sections of this manual.

The sample simulation presented shocks the INTERMOD model with an increase in US government expenditure. The increase is equivalent in size to 1% of US GDP for an 8-year period. The shock starts in 1988 and becomes zero in 1996. This shock is fully specified in the start-up file **DECK.US**. To run the simulation, follow these commands:

1) Set SIMULATOR up in your hard drive as specified in the preceding section of the manual
2) type **CD C:\SIMULATE**
3) type **SIM**
4) At the prompt for a start-up file, press **<ENTER>**
5) In the set-up menu, type **4** to set up the run
6) Type **N** to avoid saving the results for a control
   Type **Y** to read the control solution results
   Type **N** to avoid executing a policy routine
   Type **N** to avoid exogenizing sectors
   Type **N** to avoid endogenizing sectors
   Press **<ENTER>** on the # of iterations option
7) Once again in the set-up menu, type **8** to simulate
8) SIMULATOR runs the simulation shock
9) When the simulation is done, press **<ENTER>**
10) In the set-up menu, type **9** to access the data routines
11) In the data routines menu, type **3** to access the exogenous variables
12) In the exogenous simulation data menu, type **SHOW**
13) At the show format prompt, type **2**
14) SIMULATOR prints out on your screen the level of US government expenditure causing the shock
15) Now type **QUIT** to return to the data routines menu
16) Once back in the data routines menu, type **2** to access the endogenous variables
17) In the endogenous simulation results menu, type **SHOW**
18) At the show format prompt, type **6**
19) SIMULATOR displays on your screen the changes caused by the shock in several key variables
20) Now type **QUIT** to return to the data routines menu
21) Once again in the data routines menu, type **7** to return to the main set-up menu
22) In the set-up menu, type **10** to exit SIMULATOR
23) Type **N** to avoid saving your changes

IV An Overview of Simulator

The purpose of this section is to present the basic organisational logic behind SIMULATOR, as well as expand upon the HELP facilities and the data, model and default files contained in SIMULATOR. This should provide some insight into the operation of SIMULATOR and assist in running simulations.

SIMULATOR's built-in HELP facility

At most stages of SIMULATOR a HELP facility has been included. If you are not sure exactly what the program expects of you in the form of input, or if you cannot remember what the input options or format styles are, type **HELP** as an input and press <ENTER>. SIMULATOR will respond with a brief description of the input desired at this stage, as well as a listing of the various options available (if any).

A number of causes ranging from corrupted files to bad data input by the user can cause an irrecoverable error. In these circumstances the PC operating system will take over and return the system to the DOS prompt, leaving you to start over from scratch. When this occurs you should make some effort to determine the cause of the error before attempting to repeat the simulation. The error messages that are displayed will always include a 4 digit number preceded by a letter code as well as a brief description of the problem. If the description does not provide sufficient information, a more detailed outline of the problem can be obtained by referencing the error code in an appendix of the Microsoft Fortran Reference Manual.

Under certain circumstances you may also lose the start-up file when the system returns you to the DOS prompt on recovering from an error. For this reason it is advisable to keep a spare copy of your important start-up files.

SIMULATOR'S structure

The basic organisational concept of SIMULATOR is perhaps best described as a full file storage approach. This refers to the fact that all the relevant input values, parameter settings and data involved in setting up, storing, executing and printing out a simulation are fully assigned to various files outside the SIMULATOR command file itself. The SIM.EXE program contains the SIMULATOR operation commands and the model equations, but cannot be run without files specifying the variable listings, historical

data values, output storage locations, control solution values, etc. Although this may initially seem confusing, in practice it greatly speeds up and facilitates the processes of setting up or altering simulations, reexamining simulation results and running multiple simulation comparison exercises.

The parameters of a SIMULATOR operation are contained in the start-up file. This file specifies the names of the database file, the file containing the control solution, the endogenous and exogenous solution files, the output file and others. It also stores the values of the time period of the simulation, the maximum number of iterations to be used in solution and any other necessary parameters. An existing and complete start-up file must be given when SIMULATOR starts, or the program is unable to continue. You are not limited to the particulars of that initial start-up file for your simulation; all of the various specifications can be changed.

Operation of SIMULATOR is conducted through a series of menus. The main menu is enabled after a starting start-up file is specified. It provides access to a default files menu; a command for specifying the time frame of the simulation; a display title command; a menu of the run characteristics; a menu for altering the solution parameters; a start-up file retrieval command; a command to save your changes in a new or existing start-up file and the data, coefficients and residuals routines menu. By accessing these various menus and commands SIMULATOR can perform whatever simulation you desire. These specifications can be stored in a new or existing start-up file, which, when retrieved in the future will exactly replicate the simulation specified.

Once all the items of the desired simulation have been specified, the 'simulate' command can be executed. Following completion of the simulation, the various data routines will allow you to view or print the results of your simulation.

A brief description of some files

Initially when you installed SIMULATOR you loaded a series of files onto your hard disk. These files are the basic starting files for SIMULATOR, containing the data and values required for simulation runs, as well as the command structure and the model equations.

SIM.EXE is the operational SIMULATOR program. This file contains both the commands of SIMULATOR and the compiled equations of the INTERMOD macroeconomic model. The .EXE file is the executable image of the FORTRAN program that comprises the model code simulation program. It will require access to outside files

for the data, variable listings and the defaults it requires to operate.

The IODATA file is the "historical" database used by SIMULATOR. Not only exogenous and endogenous data series, but also the residuals series, are included here. The data are historically accurate from 1960 to 1985, and are steady growth scenario extrapolations from 1986 to 2020. The exogenous and endogenous series for the historical period are those of MULTIMOD's database, used with the mainframe TROLL version of INTERMOD.

V11VARS is the file listing all of the model's variable names. It provides the .EXE file with the endogenous variables, the exogenous variables, the residuals and the coefficients. It also contains the starting coefficient values of the model.

The G7xxx.FOR files are largely illustrative. They are the uncompiled FORTRAN format equations of the model, sorted into their respective country blocks. The .FOR files have been included to allow you to examine more closely the equations of the model and for their potential use in changing the model structure. However, SIMULATOR does not access or use these files. It has the equivalent equations listed and compiled within its .EXE file.

The default control solution values are contained in the CONTROL file. As with all SIMULATOR's control files, only the endogenous solution values of the control solution are stored here. The exogenous variable settings are contained in the exogenous variables file specified by the start-up file which was used to create the control solution. In CONTROL's case, this start-up file is DECK.US.

SIMULAT.D is a small, single-purpose file which contains the name of the start-up file specified as the default. SIMULAT.D is not necessary for the running of SIMULATOR and will create itself once a start-up file is input as default. The start-up file initially stored as the default is DECK.US.

DECK.US is the default start-up file provided with SIMULATOR. As mentioned in the discussion of SIMULATOR's structure, a start-up file is required to even start SIMULATOR. DECK.US contains parameter settings corresponding to those of a simulation of a shock to US government expenditure. You do not need to run this particular shock or even retain any of its parameter settings. Once SIMULATOR is running, you can and should change the file location specifications and default values. Note that once a new file name is specified as a file location for a simulation, that file is automatically accessed or created by SIMULATOR.

V Example #1: A US Money Supply Shock

This section presents a sample shock in a step-by-step format. Following through with this shock and the other example simulation on your own PC should familiarize you sufficiently with SIMULATOR's operation that you will be able to perform shocks of your own afterwards.

Starting SIMULATOR

In this example we will simulate a positive shock to the US monetary base of 3%, sustained over 6 years. To start SIMULATOR simply type **SIM**.

SIMULATOR will prompt with a "?" for a start-up file. The start-up file named in rounded brackets is the default start-up file, which will be used if no other file name is specified. In our case, DECK.US is the only start-up file we presently have available for use, so we will have to start with it. Either type in **DECK.US** or simply press **<ENTER>**, since DECK.US is also the initial default file.

If in future runs you enter a start-up file other than the default, SIMULATOR will give you the option of resetting the default to this new start-up file.

Once a valid start-up file is given, SIMULATOR presents you with the main menu, shown in Figure 1. The various menu items can be accessed in any order, so long as **SIMULATE** is not entered until after the entire simulation is ready to run. For the sake of simplicity we will deal with the menu fields in numerical order.

FIGURE 1

Set-Up Menu

Option	Function
1	Update default files list
2	Set simulation start and duration
3	Update display title
4	Set up run characteristics
5	Set simulation parameters
6	Save changes in start-up file
7	Read a start-up file
8	Simulate
9	Data routines
10	Exit

Update default files list

This sub-menu is used to specify the names of the files to be used for output storage and input retrieval. Enter the appropriate file name desired for each of the listed functions. If you press **<ENTER>** on a file name, it will keep the name presently specified. If you make a mistake, specify the rest of the names and then re-access this menu to correct your error.

File 1 is the file which stores the values of the exogenous variables used in the last simulation specifying this file. Previous simulations' solution values stored in this file are overwritten when a new simulation is completed. You should therefore be sure to rename this file if you plan to re-run or re-examine a particular simulation at some future point in time. For our example, name this file as **USM.EXO**. A handy convention for keeping track of shock files is to use a common filename for all the files used in a simulation, with different and standardized suffixes to specify their purpose. Thus USM denotes this file as part of a US money supply shock, while .EXO denotes that this is the exogenous data used for that shock.

File 2 is similar to File 1, except that it stores the endogenous solution values. This file will be named **USM.END** in our example, thereby maintaining our file-naming convention.

File 3 is a scratch file, used by SIMULATOR to store the previous type II iteration when a consistent expectations structure is used. Its name is of no relevance to our purposes, so leave it unchanged.

File 4 is the file used to hold the control solution values, to be used in calculating {shock-control} format results.Only the endogenous variables' values of the control solution are stored, so it is important to know what exogenous variable values were used to build a particular control. We can use the CONTROL control solution file, so leave this file specification as it is. When creating a new control, the suffix .CTR may be useful to assist you in managing your control files.

The file numbers 5 and 6 are reserved by FORTRAN for screen access, so the next file specification is for File 7. This is the file which contains the full historical data base that is used. IODATA is the data base that has already been coded for use with SIMULATOR, so leave this file specification as that. You will rarely, if ever, have occasion to change to another data base.

File 8 is the file which has the listings of the variable names and the coefficient values. V11VARS is the default file and is the one we wish to use. This file specification would only be

set at something other than V11VARS if you had made a change in the model structure itself.

File 9 is the file which will hold any output you may want hard copies of. When the simulator is first started the output sent to this file will overwrite previous data that may exist in file. Subsequent data that is sent to the file from within the same simulation run will be appended. If the file that is given in the list contains results that you want to keep, then you should change the filename at the beginning of a simulation. Maintaining our file name convention, enter **USM.PRT** as the specification for this file, with .PRT being our notation for a printout file.

Files 10 and 11 are not relevant to us, and File 12 is a scratch file similar to File 3, so this entry can be left unmodified.

Set simulation start and duration

This is a brief one-line command for setting the start date and duration, in years, of a simulation. If the time period shown when you access this feature is acceptable, just press **<ENTER>** to default to this value. This example will begin in 1988 and run for 8 years, so enter **1988 8**.

If you are looking at old simulation results, the time period setting must match the original setting active when the results were obtained, or SIMULATOR will not be able to access the earlier results. As well, you should ensure that the time period setting for your control solution overlaps fully with that of your simulation, or else you will be unable to execute any features of SIMULATOR which require the control file. This time period setting is the default time period used in most of the data, coefficient and residuals routines.

Update display title

This, too, is a one-line command, in this case used to allow you to enter up to 80 characters as a title for your simulation. The title will be printed above each output section and allows you to have meaningful labels on your simulation results. If you select this option you must provide a new title since the old one is automatically erased as soon as you enter the option.

Set up run characteristics

The run characteristics are a series of options which detail to SIMULATOR how the data and simulation operations should be manipulated. The first option is used to run a simulation to set a base-case scenario (a control). The various specifications entered are used in a simulation, whose endogenous variables' values are stored under the file name given for control storage in the file defaults menu. Since only endogenous variables are saved, all changes to exogenous variables in the base-case solution will not be saved and therefore must be re-entered with the simulation each time it is run. Take, for example, a base-case solution which is created with an increase in government spending. Any subsequent simulations using the base-case solution which remove the changes in government spending from the list of changes to variables will have the effect of shocking the system by a negative change in spending. Since we already have a valid control for use with our example, enter **N** or press **<ENTER>** on this option.

The second option is used to access a previously-created control solution, to allow you to analyse the results of the simulation using (shock-control) output formats. This a particularly useful format for the analysis of policy effects, so enter **Y** at this option to give us the chance to look at our example results using this format.

The option to execute a policy routine allows you to set up a complex, many variable shock in an outside DOS file, then import the file as a policy routine to be used in simulation. This example is comparatively simple and we will not need to set it up outside SIMULATOR; therefore type **N** or press **<ENTER>** here.

The 'exogenize sectors' option gives you to ability to remove one or more of the country blocks from the model interactions. When a particular sector is exogenized, its equations will not be executed, therefore the variables of that sector will remain at the database value. This option is useful if you wish to exclude international feedback linkages from your simulation. Our example will be run with all international linkages intact, so leave this option at **N**. The 'endogenize sectors' option is used to switch back blocks that have been exogenized in earlier simulation runs.

The final 'run characteristics' option, setting the maximum number of iterations, is primarily used to provide a point at which we can conclude that the model is not converging and the simulation should terminate. It is, however, in a secondary function that it is most useful. By setting the number of iterations to a low value, SIMULATOR will stop and you will have a snap-shot of model variable values at that point. If the model is 'exploding' under a given set of shocks, this tool will allow you to determine which variable(s) is/are causing it to become

unstable by taking these snap-shots at different points. For most cases, including our example, **99** is a good starting number for this option.

When the model is forward looking a further prompt will appear in this list requesting the maximum number of Type II iterations. This refers to the number of iterations described in the Fair-Taylor algorithm to achieve a consistent-expectations solution.

Set simulation parameters

This menu is used to specify the solution criteria that you wish operative when SIMULATOR solves the model. The first question that appears is actually asking if you want to run the adaptive expectations version or the rational expectations version. By setting the value of the forward-looking coefficient to be 1 you will have selected the rational expectations version, whereas setting it to 0 will select the adaptive expectations version. Selecting a value between 0 and 1 will result in a hybrid of the two expectations scheme. The basis for this selection is found in the form of the expectations variables coded into the model. The equations are coded in the form:

```
AA = FORWRD*LEAD(1,AA) + (1-FORWRD)*f(adaptive
     expectations)
```

so that selecting the value of forward-looking coefficient (FORWRD) selects a convex combination of the two expectations schemes. If a non-zero value is selected for the coefficient, SIMULATOR will use a more complex set of data structures and a different solution technique to solve the model. This technique has a large impact on model solution times. The convergence criteria as specified in DECK.US are sufficient so the second question can also be answered by typing **N** or pressing **<ENTER>** to leave them as they are. The prompt for the date at which to begin using the previous period's solution, as opposed to the database or control solution values for the kick-off values for a given year, can significantly affect the running of a simulation. Although it does not normally provide any advantages in running a model, it has been included to help in possible debugging of the models (you should be aware that when experimenting with this option it may have the opposite effect of destabilizing the model since the previous period's solution for endogenous variables combined with the current period's exogenous data may constitute a destabilizing shock to the model). Pressing <ENTER> will retain the current setting of **2020** which will be sufficient for our simulations.

Save changes in a start-up file

This command will save all the file settings, run characteristics, and data, coefficient and residual alterations in a start-up file. You can either overwrite the presently active start-up file or you can specify a new start-up file. Once a simulation is entirely specified and ready to run, it is usually a good idea to save the specifications in a start-up file before you simulate.

We have not yet performed any data alterations for our sample shock, so it is not really worth while saving the specifications yet. Once you have finished detailing the data revisions, you should access this command and save this shock in a new start-up file called **USM.SHK**, with the .SHK being convenient file notation for a start-up file.

Read a start-up file

Accessing this command allows you to change the simulation specifications to those of an established start-up file other than the start-up file presently active. This command is particulary useful when you have erroneously made some changes to the default settings and you would like to start over again.

Simulate

This command should be accessed only after the entire simulation has been specified accurately, as it causes SIMULATOR to begin solution of the model. The title of the simulation, certain of the simulation settings and any alterations to coefficients, data series or residuals are listed; then the model is solved. Once simulation is complete, pressing the **<ENTER>** key sends you back to the main menu of SIMULATOR. Do not yet access this command, as you have not yet detailed the data revisions necessary for our example shock.

Data routines

The data routines are perhaps the most interesting for three reasons: the first is that they are the means by which alternative policy scenarios can be simulated for counter-factual analysis, the second is that they permit equation parameters to be manipulated so that model properties can be studied in relation to parameter values, and the third is that they allow the results of the various simulations to be examined in a number of ways. The following paragraphs will focus on the facility to change exogenous

data for the purpose of implementing policy. Later sections will deal with the remaining functions of the data routines.

The historical data values display provides a means of viewing the values of the data base. The data base series are termed historical, but in actual fact span the time period from 1960 to 2020. The data are taken from SNA sources for the 1960 to 1985 period, and are forecast based on steady growth projections thereafter. The exogenous data that appear in this display can be altered for a given simulation without changing the value in the data base. These changes can be stored in the start-up file so that while they are temporary changes, they can be used at any time by accessing the start-up file with the changes.

In this example, we want to increase the US money supply by 3%, so it would be interesting to find out just what the data base series of the US money supply target is. Therefore, type **1** to view the data base series. Now type **X** to view the exogenous series of the data base.

The screen clears, and you are presented with a display similar to that in Figure 2.

FIGURE 2

Historical Data Display

```
1:UG          2:UBRAT       3:UDLTA       4:UGXOG       5:UMULT       6:UPIBR
    7:UCTRT       8:UXOIL       9:UPRDL      10:UTXCD      11:UMT

Enter variable index and mnemonic to update

 ?
```

The numbered variable names across the screen are the names of those variables that are presently selected. Any command that we issue to perform some function with the data will affect only the variables in the list. Typing the number preceding a selected variable clears that variable from the selection list, while typing the number followed by a space and then a variable name causes that variable to appear on the corresponding part of the data display. Up to 120 variables can be selected at any one time.

In addition to altering the selection list, there are other commands that can be entered. They are:

REFRESH - this updates the selection list and displays it on the screen.

NEXT - this command displays the next page of selected variables (up to 60 can appear on the screen at a time).

BACK - this moves the selection display back to the first screen.

DISPLAY - this command scrolls through the full list of variables in a particular category. This is useful if you want to see what variables are available for selection and what their mnemonics are. **<Ctrl><S>** will freeze the screen and any key will start it again.

RANGE - this enters a range of variables, based on the numbers assigned in the DISPLAY listing, into the selection list, remember that only 120 variables can be selected at one time.

SHOW - this gives a screen display of data series for a specified time period. Either all the variables in the category or only the selected variables can be displayed.

PRINT - this acts in the exact same manner as SHOW, except that the output is sent to the print file instead of onto the screen.

QUIT - this returns you to the main data routines menu.

For this example, we wish to see the data base values of UMT, the data series giving the values of the US monetary base target. Therefore:

1) Type **RANGE**

2) Now type **1 1** - this clears all the previous settings in the selection list, and only adds the variable assigned the DISPLAY listing number 1 onto the selection list.

3) Type **1** - this clears the selection list

4)	Type **REFRESH**	- our selection list is now shown to be empty
5)	Type **DISPLAY**	- the full listing of exogenous variables is scrolled (lets you find or verify the name of a variable)
6)	Type **1 UMT**	- this selects the US monetary base data series
7)	Type **REFRESH**	- this will verify, if you wish, that the previous selection has been entered correctly.
8)	Type **SHOW**	- if we want to see the values of UMT
9)	Type **1988 8**	- to specify, e.g., that the series is to be shown for eight years starting in 1988
10)	Type **2**	- our selected series is shown on the screen
11)	Type **QUIT**	- now we have seen the data base levels of UMT, so we can go on to the other data routines

The data routines for examining the endogenous simulation results and the exogenous simulation data can only be used after a simulation to view the effects of the simulated shock. There is therefore no point in accessing these routines at this time. Furthermore, in this example we are not changing any of the model's coefficients or any of the residual values, so the only data routines we still need to operate before simulation are the ones for revising data. To move to this menu, type **5**.

You now see a screen that is very similar to the one you saw when examining historical data series. In fact, all the data routines menus are structured similarly, and they generally respond to the same set of commands. To change or add exogenous variable shocks, all that you need to do is add the variable you wish to revise to the selection list in this menu.

We want to revise the variable UMT to create our shock, therefore:

1) Type **1 UMT** - this sets UMT as a variable selected for revision

2) Type **1988 8** - we want to shock UMT over the full 8-year period of our simulation

3) Type **%** - since we want to increase UMT by a percentage, this is the logical format to choose for our revision

4) For each year, type **3.0** - it is important to include the decimal point, or else SIMULATOR will misinterpret the change desired

5) Type **SHOW** - this allows us to check that only the variables we want to revise have been revised, and by the correct amount

6) Type **QUIT**

We have now fully specified our shock, so exit the data routines by typing **7**. Once again we return to the main set-up menu. Type **6** and enter **USM.SHK** to save this example shock in a file of that name. With all systems set to go, type **8** to have SIMULATOR solve the model. The screen should clear and then the procedure detailed earlier in the description of the Simulate command should commence. Once the simulation is completed, press **<ENTER>** to return to the main set-up menu.

Our example simulation of a +3% shock to the US monetary target is now complete. The results of this simulation are stored in the files we named as the exogenous and endogenous solution files, namely USM.EXO and USM.END. The next section will detail the various ways in which you can examine these results.

VI Examining Results

There are two basic output options available with SIMULATOR. To view data on your PC's monitor the SHOW command is used, while the PRINT command enables you to direct output into a DOS text file for later printing.

Regardless of whether you are using SHOW or PRINT, you should be careful to ensure that the start date and duration you specified in the set-up menu are compatible with the dates of the solution files you are examining. When you are looking at the database values this will not be a problem, since the start date and duration are irrelevant for displaying those data. If your dates are not properly set the system will inform you of your error and you must correct it before proceeding.

SHOW

The SHOW command is helpful in all of the data routines. In the coefficient adjustments menu, the data revisions menu and the residuals revisions menu typing SHOW will cause SIMULATOR to display the particular revisions presently active in that category. The revisions are shown in a continuous scroll; again, **<Ctrl><S>** freezes the scrolling and any key resumes it.

Using SHOW in the historical data values menu gives you a return prompt asking which display option you want, 'all level' (1) or 'selected level' (2). The 'selected level' option will display the database values, over the specified period of time, of only those variables in the selection list. The 'all level' option will display the database values of all the variables in that category, (i.e. all the exogenous variables, all the endogenous variables or all the residuals).

In the exogenous simulation data menu, used to examine the values of the exogenous variables used in a simulation, SHOW works exactly the same as it does in the historical data values menu. However, in the endogenous variables menu, SHOW has considerably more output options. The choice between seeing the selected variables only, or all the variables, remains. In addition to this, you can also choose to focus only upon the differences between the simulation results and the control solution values. The differences can be displayed in either absolute terms or as percentages of the control solution values. Finally, you also have the option of varying the output format with each series, through the use of the mixed mode option. This option will prompt you at each series for the output format you desire for that variable. Mixed mode output can only be used in conjunction with the option to display selected variables.

When you have specified your output options, SIMULATOR displays your simulation title and the category of data being shown, then scrolls the values of the data series.

PRINT

The PRINT command works in exactly the same manner as the SHOW command and has the very same options for output. The difference between the two is that while SHOW displays output on the screen, PRINT displays nothing on the screen, but instead sends the output requested directly to the print storage file. Recall that the first PRINT command of a simulation overwrites everything previously stored in that print file, while subsequent PRINTs append their output to the file.

PRINT is used to obtain hard copies of data values. A suggested procedure to be used in creating these hard copies of data is this:

1) After a simulation is complete, go through the various data routine menus and use the SHOW command and selection lists to sort out which data series you want printed.

2) Use consecutive PRINT commands, specifying your desired output options, to send copies of the data series into the print storage file.

3) Exit SIMULATOR and print the print storage file, using either a word-processing package or the DOS command PRINT.

VII Example #2: An Oil Price Shock

In Example #1 we went through the basic features of SIMULATOR and conducted a simple shock. Now in Example #2 we will explore several of SIMULATOR's other modelling features, namely creating control solutions, altering coefficients and running a simulation using the consistent expectations scheme. We will be chiefly interested in the oil price in this exercise since the equation for the price of oil provides a good basis for our experiments.

Resetting coefficients

Start SIMULATOR by once again entering **SIM** in DOS. **DECK.USR** will serve as the initial start-up file for this run. The world oil price in INTERMOD is represented by the endogenous variable POIL. The equation for POIL is simply a weighted average of the GNP deflators of the sectors in the model multiplied by an exogenous real oil price index, RPOIL. The relationship between the two variables is :

POIL=RPOIL * [the total product of each sector's
((PGNP deflator*exchange rate/
1980 exchange rate)**sector GDP weight)]

where the sector GDP weights are coefficients, denoted as UWT, JWT, GWT, FWT, KWT, IWT, CWT and SWT.

To keep this example simple we will set POIL to 1 for the entire period. Since this variable is a nominal index we will have the price of oil falling in real terms. Since RPOIL is exogenous it is easily set at 1.0 for our solution period. To proceed, enter the data routines menu and then the data revisions menu. Next, clear the selection list (refer back to Example #1 regarding how to do this) and add **RPOIL** to the selection list. Specify **1980 16** for the time period of the revisions and choose the **REPLACE** option for the revision format. Now enter **1.0** for each period of the revision. You can check your revisions using SHOW if you so desire.

Having set RPOIL to 1.0 we now need to force POIL to the same value. To do this we will set the coefficients of the POIL equation to 0.0, thereby effectively reducing the equation to:

POIL=RPOIL=1.0.

This is effected by the following:

1) Enter the coefficient adjustments menu

2) Type

1	**UWT**	**0.0**	- This sets all the
2	**JWT**	**0.0**	coefficients at 0.0
3	**GWT**	**0.0**	
4	**FWT**	**0.0**	
5	**KWT**	**0.0**	
6	**IWT**	**0.0**	
7	**CWT**	**0.0**	
8	**SWT**	**0.0**	

3) Type **QUIT**

Note that if you try to see these revisions by entering the coefficient name alone, you will only see the old coefficient values. This is because coefficient adjustments do not take place until the simulation is actually run. Furthermore, the adjusted values become the default values for the rest of the SIMULATOR session. Exiting and starting a new SIMULATOR session will reset the coefficients to their original values.

Creating a control solution

Now that we have set the price of oil at a constant nominal level, we can proceed with the analysis by either one of two methods. First, we can create a control solution with the constant nominal price and later run the historical solution to obtain a comparison. Alternatively, we could use the historical solution as a control and run the current setting as a shock to history. In this example, we shall explain the steps involved in the first of these two approaches.

Tackling the set-up menu in numerical order, we move first to the default files menu. POIL seems an appropriate filename for this simulation exercise, so set File 1 to **POIL.EXO**, File 2 to **POIL.END** and File 9 to **POIL.PRT**. Also, since you will have to create a new control solution, set File 4 to **POIL.CTR**. The rest of the files can be left at their present defaults.

Set the simulation start and duration for **1985 31**. This will allow our control solution to cover a simulation period from 1985 to 2015 which will provide sufficient data for the 3 lagged and 28 future expected periods that may affect our solution. The run characteristics should be set to save the results as a control solution, but leave the policy routine execution, sector exogenization and sector endogenization options at **N**. The maximum number of Type I iterations can be left at **99**. As well, the maximum Type II iterations can be left at 500. You can also leave the simulation parameters at their defaults.

The main area of differences in running the adaptive and consistent expectations models is found in the main menu option 5. You will now respond **1.0** to the prompt which asks you to set the value of the forward-looking coefficient. With the second prompt you can change the convergence criteria used in both Type I and Type II iterations[2]. The algorithm that SIMULATOR uses can begin with loose convergence criteria on Type I iterations which are tightened to the final convergence criteria. You have the ability to set both the initial criteria and the rate at which they will step to the final criteria. SIMULATOR uses two tests to check convergence of a model. The first is a percentage change test between iterations and, failing this test it will try a units test. If we were creating a control solution with no shocks we could start with the final Type I convergence criteria since the final

---

[2]A concise explanation of the extended path algorithm and the solution technique implemented in this system is provided in Section IV of Helliwell, J. F., G. Meredith, Y. Durand, and P. Bagnoli (1988), "Intermod 1.1 A G-7 Version of the IMF's Multimod", Ottawa, Department of Finance, September 1988. A short outline is also given in a later section of this guide.

solution will be reached[3] quickly. Since in our case we are creating a control solution with a shock, we will use the loose default setting.

The next question allows you to change the number of periods for which the system will actually conduct Type II iterations testing. Experimentation has found that we do not have to solve for tight Type II convergence for all forward periods to obtain a good approximation of the final results. If our interest is primarily in the six periods following 1988 then the default will be sufficient. However, if we are interested in taking a close look at later periods then this value should be increased.

The remaining step will be to simulate the model to obtain the control solution (entering 8 in the main menu). Before doing this, however, we will need to save the changes we have made in a new start-up file. Select menu item 6 and respond **N** to the first question which asks if you want to overwrite the current start-up file. Now enter **DECK.POL** as the file into which we will save the parameters. When you are back in the main menu, select menu item 8 to conduct the simulation. Once we have completed this step, we will then be in a position to try utilizing the control solution to conduct a shock minus control analysis. When the simulation has completed (it may take some time), we will want to obtain the historical solution for comparison. The coefficient changes that we have made will remain in place until we make further changes to reset them to previous values. Since it will be easier to simply exit the simulator program and restart with the previous values, we will proceed in that manner. When restarting the simulator, the start-up file should now be **DECK.POL**. The Type I convergence criteria should be tightened and the changes to the coefficients and exogenous variables should be removed from the list. The final step will be to set the simulation period to **1988 20**, after which we can select menu item 8 to conduct the simulation. The previous steps discussed regarding examining solutions can now be carried out to look at the results.

Using a consistent expectations structure

SIMULATOR uses a Gauss-Seidel algorithm to solve the model. In the adaptive expectations version, expectations are formed for previous values; therefore once the solution iterations for a single year have converged sufficiently to meet the specified criteria, (Type I iteration criteria), SIMULATOR stores the last

---

[3]Since the residuals are in the equations the final solution should be reached in one iteration. However, small errors in the data in the later years cause the system to conduct a number of iterations before finding the historical solution.

iteration's solution set as that year's final solution set. The consistent expectations version uses a Fair-Taylor algorithm to obtain forward-looking expectations solutions, so the final solution set for a particular year must not only be internally consistent, but it must also be consistent with the future values upon which expectations are being formed. Since these future values are themselves part of a future year's solution set, SIMULATOR must not only conduct Type I iterations upon each year until it arrives at a solution set with all that year's variables consistent with each other, but it must also conduct Type II iterations upon all the years' solution sets simultaneously to ensure that expectations are in line with the actual future values they predict. The criteria which determine when successive iterations have converged sufficiently to ensure solution consistency can be set in the simulation parameters menu.

The default settings of the convergence criteria have been tested and are generally efficient at producing reasonable results without spending time meeting overly tight criteria. There are separate criteria settings for Type I and Type II iterations, with the criteria expressed in the primary form as percentages. If the percentage criteria are not met, then there are also unit criteria which are checked.

The simulation parameters menu also has an option for the setting of the convergence horizon, i.e. the number of years in the future for which Type II convergence will be strictly enforced. The longer this convergence horizon becomes, the further forward the true converged solution will be found, and the longer the solution time for the model.

The option for setting the year at which SIMULATOR is to begin using solution values as starting points has already been discussed in Example #1. One final note remains regarding the Fair-Taylor algorithm and the simulation parameters menu is that it has been shown that the efficiency of the Fair-Taylor rational expectations solution algorithm may be improved if all the years of the simulation are solved simultaneously as a single equation block. SIMULATOR does not perform this process, but it does approximate it by having initially loose Type I criteria, then gradually tightening these criteria. This is why the simulation parameters menu has separate criteria settings for starting and finishing Type I criteria, as well as for the step sizes between these two.

VII Example #3: Changing Exchange Rates

This final example is intended to demonstrate the system's capability to alter residual values. We will use this

feature along with the change coefficients feature to exogenize the exchange rate equations and then to residually set the exchange rate.

Revising residuals

Using Canada for our example, the equation for determining the bilateral exchange rate has the form:

CER = DUMEXG*(equation) + RCER

where RCER is the residual and DUMEXG is a coefficient. By default the value of DUMEXG is 1.0. If we change that value to 0.0 we will cut off the equation which normally determines exchange rates and leave it to be set by the residual. Changing the residual value is carried out in much the same way as changing an exogenous variable. By selecting menu item 6 in the data routines you will be in the menu to change residuals. As in the exogenous variables changes menu, you will be required to enter an index followed by a variable name. SIMULATOR will then ask for a start year and number of changes after which you will be required to manually enter the new values of the residual.

The exchange rate for other countries has exactly the same form as the equation for Canada. This means that when the Canadian exchange rate is exogenized all the other exchange rates will also be exogenous thus requiring you to set the residual for all countries' exchange rate equations.

PART II - Model Builder's Guide

I Introduction

This manual is designed to assist in modifying existing models and creating new models with the PC simulator package. A large part of this manual describes how to assemble a model for use with the simulator. However, since many users will simply want to modify an existing model, a separate section (section VI) is provided to outline the procedure for revising an existing model. Section VI assumes that the model variables are to remain unchanged and that it is only the form of an equation that will change. If a variable is to be redefined or added to the model, then most of the steps to build a model will have to be repeated.

The Fortran compiler used to build models with this simulator is the Microsoft Fortran compiler. Although theoretically this compiler should be compatible with other compilers, it is our experience that this will likely not be the case. Thus it is highly recommended that users obtain a copy of the Microsoft Fortran compiler. The compiler will produce an intermediate object file (a file with a .OBJ extension) when the /c switch is set (section III.1 demonstrates the entire command line to compile a program). This object file is LINKed to other object files to produce the executable file that is invoked in DOS to run the program. The simulator and various other utility programs (discussed later) are distributed in object file form so that the user will only have to compile his own programs and link the object files produced with the distributed form of the simulator.

The organization of this manual is designed to present the simulation package in three units comprising:

1) coding a model
2) providing data for the model
3) building the residuals for the model

The final sections provide summaries and some helpful hints on using the system. Some people may feel more comfortable having a general view of the steps before delving into the details of each section. These users should read sections V and VI before proceeding to section II.

II Model Building

Each model is a set of Fortran subroutines that are activated by the simulator. When the model is simulated, it must obtain data from a database file and associate those data with the variables that are used in the model. This link between the data and the variables used is established in a file called EQNAME.FOR. When the simulator is instructed to run the model it will have to know which subroutines comprise the model. These subroutines can be grouped into named sectors to provide a conceptual organization for the model. The link between the subroutines that form the model and the named sectors which are activated by the simulator is established in the program called ORDER.FOR. Sections III.1 to III.3 outline the modifications to these files that are required to properly set up the links to build a model.

II.1 Structure of Model Coding

The structure of the model coding is straightforward and requires only a minimal knowledge of Fortran coding conventions. The model itself is built as a set of Fortran subroutines which are activated by the simulator through CALL statements. A model can be comprised of a number of subroutines or it can be built into one subroutine. The most efficient use of subroutines is to associate a sector of the model with a particular subroutine. For example, in an international model each subroutine may represent a country block. This technique allows a sector of the model to be made exogenous by simply omitting the CALL to that sector's subroutine. In a model where the country blocks are large and cannot be fit into a single subroutine, the simulator provides a means of grouping several subroutines into a sector. This is discussed further in section II.2.

The basic skeleton of the subroutine is as follows:

```
SUBROUTINE subname(NTIME,YZ)
$INCLUDE: 'EQNAME.FOR'
equations ...
RETURN
END
```

The coding of the equations must adhere to Fortran syntactical conventions. Appendix II provides a listing of some sample equations coded in Fortran.

Along with the standard set of functions provided by Fortran, two other functions are provided to access data. These are the LEAD and LAG functions which allow references to be made to data other than that of the current period. Use of the LEAD function is only available when the model is forward-looking. An important variable used in conjunction with the LEAD function is the FORWRD coefficient. The value of this coefficient is set by the user at the time the model is run. The variable will be set to 1 for a forward-looking model and 0 for an adaptive model. However, intermediate values are also permitted by the system so that something less than perfect foresight can be assumed to be the rule of agent expectations. The simulator uses a different set of data structures which require substantially increased solution time when running a forward-looking model. These data structures are used whenever FORWRD differs from zero so that if the user is primarily interested in the adaptive expectations scheme, the value of FORWRD should be set to zero to avoid utilizing the forward-looking data structures. The model in Appendix II makes use of the LEAD function and provides an example of combining the expectations schemes.

The functions LAG and LEAD which are provided in the simulator cannot take functions as arguments. A statement of the form:

```
A = LAG(1,2*B)
```

will return a nonsensical value. However, the statement:

```
A = 2*LAG(1,B)
```

will return the correct answer.

The subroutines are also provided with a time index called NTIME which is equal to 1 in 1901. If some section of coding is to come into effect in a particular year, an IF statement can be used to generate the proper structure, eg. to key on 1985 the statement would be written as:

```
IF (NTIME.EQ.85) THEN
 equations ...
 ENDIF
```

At the present time the data structures built into the PC system impose limits on the building of a model. These limits are:

```
# of endogenous variables - 600
# of exogenous variables - 600
# of coefficients - 1000
# of residuals - 400
Max. forward reference of data - 5 periods
```

Max. lag reference of data - 5 periods

These limits can be changed but they will affect the size of the programs and thus may cause the program to not fit into the memory space provided by the PC operating system.

Once the model subroutines have been written, they will have to be compiled using the Microsoft Fortran compiler. Assuming that the compiler has been installed and properly configured, the command to compile programs with is:

```
FL /AH /c name.FOR
```

Where 'name.FOR' is the file containing the subroutines. Notice that the file must have a .FOR extension. This command will create a file with an extension of .OBJ which will then be linked with the simulator program to create a model.

II.2 The ORDER.FOR Program

The simulator provides the ability to name up to 30 sectors where each sector name can be associated with a large number of Fortran subroutines. All of the work required to set up sectors is performed in the program ORDER.FOR. Thus the instructions given in this section will implicitly refer to this program. To set up a sector, a name (at most 6 characters) must first be assigned to an element in the array SECTRS. This is done with a statement equivalent to:

```
SECTRS(xx) = 'mysect'
```

The name that is used in the array is used by the program to establish a connection to the subroutine that the user creates. The manner in which this is accomplished is through the use of a logical array which is referenced according to the index chosen by the user for the name array. In other words, when the Fortran subroutine is activated through the use of the statement:

```
IF (DUM(xx)) CALL subname(NTIME,Y)
```

the index xx must be the same index that was used earlier when the sector was named. Notice that since many subroutines can be called based on a given index, a virtually unlimited number of subroutines can be associated with a given sector name. It is important that the 'subname' of the subroutine that is called in the above statement must be the same 'subname' that was given to the subroutine as described in the section entitled Structure of Model Coding. A sample listing of the ORDER.FOR program is provided in Appendix III.

The order of the sectors in a model can be of critical importance in a Gauss-Seidel solution technique. With this simulator the user must establish the desired ordering in the ORDER.FOR program. This is simply a matter of placing the CALLs to the subroutines in the order in which they are to be activated. Users may change this ordering at any time and sectors can be called repeatedly before proceeding to the next sector. It will be important, however, to re-compile the ORDER.FOR program and link all the programs together again each time a change is made.

This program also requires the user to set the maximum number of periods that the data will be led and lagged. It should be emphasized that these values should not greatly exceed the length of the leads and lags actually used, if rapid model solution is desired.

II.3 Putting the Model Together

Once the equations have been coded and the changes made to ORDER.FOR the next steps will be to compile the code and create the residuals of the model. These steps are outlined in section IV. After the steps in section IV have been completed, the following step links the model to the simulator programs so that simulations can be conducted. The DOS command to link all the programs together is:

```
LINK INTER1+INTER2+SCREEN+ORDER+model programs /E
```

The system will return with a number of prompts which must be answered. The <ENTER> key can be pressed for the second and third questions but a name should be entered in response to the first question. The name entered will be the name of the program that is created and that will later be entered by the user in DOS to invoke the simulator.

III Data Structures

Internally the system uses five data structures to store data. These consist of:

(1) Endogenous data (current and lagged)
(2) Exogenous data (current and lagged)
(3) Residuals
(4) Coefficients

(5) Forward endogenous data

These data are stored in two files that are referenced by the simulator. All data other than the coefficients are taken from a database file and read into these structures. The coefficients are taken from a separate file which lists the coefficient names and values. This second file also contains a list of all the variables used in the model and forms the central variables list which is used in all stages of building and running the model. It is worth emphasizing that changes made to this file have repercussions for all model-building stages that are not automatically performed by the system. When a change is made to this file all the steps required to build a model will have to be repeated. The file has a specific format which must be maintained in order to have the file read properly. In the default files listing menu of the simulator, this file is defined as file 8.

Appendix I provides a sample of the variables file. Notice that each variable is contained within six character spaces followed by two blank spaces and each line has at most ten entries in it. The position of the items in the file can be found utilizing the column indicator at the bottom of the listing. The coefficients are entered in a somewhat different format, with each entry followed by the coefficient value.

When building a model which is not forward looking (ie. does not make use of the LEAD function) it is important to disable usage of the forward data structures and routines. Use of the forward-looking structures and routines creates a substantial overhead which is unnecessary if the model is not intended for use with consistent expectations. As was described in Section II.1, the forward-looking data structures can be disabled by setting the forward-looking coefficient to zero (main simulator menu item 5). Use of the LEAD function is only possible when the forward-looking coefficient has a non-zero value. If the model is not forward looking and the LEAD function is used, it may return a value of zero or it may return a meaningless value. A model that has been defined to be forward looking will automatically be solved utilizing the Fair-Taylor algorithm.

III.1 Technical Note

The storage structures are two-dimensional arrays where the first dimension is a variable index and the second dimension is a time index. Since Fortran stores arrays in column major form, this storage scheme will group all variables for one period in consecutive storage spaces followed by all variables for the next period, and so on. Therefore, variable names which are mapped on top of this storage space - using a COMMON statement - will

reference whatever data has been read into the array. The LEAD and LAG functions then have only to calculate some index across the groupings to obtain the required data.

IV Model Building Utilities

This set of utilities is designed to achieve four basic functions:

(1) build the file which links the data in the database to the variables used
(2) build a database in binary format
(3) build the model used to create residuals
(4) generate model residuals

To get started, a number of programs must first be linked together to create an executable main program. The command to accomplish this is:

```
LINK UTILITY+SCREEN+ORDER+(SUBS) /E
```

The user can simply press <ENTER> in response to the subsequent prompts by the system. This creates the program UTILITY which will be used to perform the tasks described in the following section.

When the UTILITY program is started it will request a start-up file. The file performs an identical function to that of the start-up file used in running the simulator - described in Part 1 - but is in a somewhat different format. The file DECK.UTL which has been provided can be entered in response to the prompt and modified as needed. As with the simulator start-up file, the modifications can be saved so they will not have to be re-entered each time the UTILITY program is used.

IV.1 Build file to link data to variables

This step is crucial for all subsequent steps in building the model. The file that was created in Section III to contain the list of model variables is read and a file is created which allows the Fortran compiler to link the data to be read from the database to the variables used in the model. The new file created will be called EQNAME.FOR. The file must be 'included' into your model subroutine as is shown in section III.1.

To build the new file requires two steps both of which are carried out from the main menu of the UTILITY program. First, select option 1 and ensure that file 8 in the files listing is the file that contains the list of model variables. Next, select option 5 from the main menu. This will immediately build the EQNAME.FOR file from the list of variables found in the file attached as file 8. Any previous version of EQNAME.FOR that exists will be overwritten by this operation.

IV.2 Build a binary database

This step builds the model database in a binary format for fast access by the simulator. Since the model's execution time is directly related to the amount of data that has to be read before the simulation begins, it is good practice to maintain a database with a starting date as close as possible to the start of the relevant period (allowing for model lags).

The program reads an ASCII database file for the necessary data and creates a binary database file. The binary file created is the one specified as file 11 in the files listing. This file will be used in the next stages when the model residuals are built. The ASCII database is in a specific format, which must be respected to allow the file to be read. The SNA database built at the Department of Finance is provided to demonstrate the format. This is contained in two files called BASE1.END and BASE2.EXO which correspond to the endogenous and exogenous data of the INTERMOD model. The source database file can be either split into a source for endogenous data and a separate source for exogenous data or it can be left in a single file. The main benefit of splitting the file is found in keeping it small enough to be edited by a standard text editor.

The program will also ask for a residuals file. If any of the model residuals have been constructed elsewhere and the user wishes to use those values, then a database file with the residuals will have to be created according to the format of the preceding ASCII database files. The header should contain the exact name of the residual as it appears in the variables listing file.

IV.3 Create model to generate residuals

Having built a model, the next step is to generate the model residuals. This will be accomplished by using a program

which analyses the model code and builds a transformed model where the left and right hand sides of each equation have been isolated so that the residual can be determined as the difference between the two. The only task that we are required to perform with the program is to provide a name for the new model that the program will create. This new file must be compiled with the Fortran compiler and linked into the utility program. If the entire model is contained in several files, then the procedure will have to be repeated for each file.

A further step is required to indicate to the UTILITY program the association between residuals and the endogenous variables, i.e. which residual variable name belongs to which endogenous variable name. This is accomplished by creating a file which contains a list (one per line) of endogenous variables. This list must match one-for-one the list of residuals as given in the variables listing file, i.e. the first endogenous variable in this file will be associated with the first residual of the residuals list. In this way the system can create the necessary relationship between the residuals and the endogenous variables they are associated with. The file created is entered as file 13 in the files listing menu of the utility program. Entering option 5 from the main menu of the UTILITY program will build the converted model.

The technique used to calculate residuals is to simply take the difference between the left and right-hand sides of an equation. However, because not all equations can be neatly split with an additive scalar residual, two special prefixes are recognized by the program as having special functions. These prefixes are placed on the list of variables described in the preceding paragraph. The first is the ampersand '&' symbol which is used to denote that a logarithmic residual is to be calculated. If the equation is of the form:

$$\log(endog) = \log(vars...) + \log(residual)$$

then the variable **endog** would appear in the list as **&endog**. The second is a percentage '%' symbol which indicates that the residual is already in the database and should be retained. Variables that use the percentage symbol should have had the residuals entered into the database when the binary database was first constructed. Appendix IV provides a sample of this file.

IV.4 Generate residuals

The most efficient manner to outline this section is to demonstrate it with an example. Assume a model has been built and placed in the file called MOD1.FOR. The necessary steps have been

carried out up to and including section IV.3 where a converted file was created and given the name MOD1RES.FOR. The necessary change in ORDER.FOR was also made (outlined in section II.2) and both programs were compiled. The next step is to generate a new UTILITY program which will be capable of creating the residuals. To do this we type the DOS command:

```
LINK UTILITY+SCREEN+ORDER+(SUBS)+(MOD1RES) /E
```

Notice that MOD1RES is enclosed in a separate set of brackets. This step creates an DOS overlay which will ensure that we keep the entire program small enough to fit into memory. If the model is large it should be subdivided into multiple subroutines and have each subdivision enclosed in its own brackets. A rule of thumb as to size is that each subdivision should have no more than 150 equations in it. The indication that further division of the code is necessary will occur when the program is started. If the program is too large the operating system will simply respond 'Program too big to fit in memory'.

The new utility program is now activated and the period for which residuals are required is also selected. Changes will also have to be made to the files listing since section IV.2 created a new database which was stored in file 11. This file (file 11) now becomes the input database for the current stage (file 7) and a new database with the proper residuals will be written to the file attached as file 11. When we begin conducting actual simulations with the model, this new file (file 11) will be the input file for the simulation (file 7 in the simulation program). The file described in section IV.3 to indicate the association between residual variables and endogenous variables should have been created and should be listed as file 13.

The final step is to set the system to create residuals (main menu option 3) and then proceed to run the model.

V Summary

This section provides a brief overview of the main features of model building. It is intended to ensure that the user has a general grasp of the manner in which the previously described sections of this manual fit together.

The model equations are written into subroutines according to Fortran coding conventions. This separation of the model into subroutines is primarily required to prevent the code from becoming too large. However, a benefit of this subdivision is that the model can be organized into sectors which can be

analysed in isolation. When the subroutines are created they are grouped together into named sectors in the file ORDER.FOR. The name given to each sector in this file can later be used to exogenize the sector. Note that each sector can contain a number of subroutines or only one subroutine, the choice being determined by the amount of code that will comprise the sector.

The declaration of variables and coefficients used in the model is provided in a separate file. This file is the source for the names of the various types of model variables and also provides the source for coefficient values. This variables listing file is created in a prescribed format which is read by a utility program to build the file EQNAME.FOR. The Fortran subroutines will use this EQNAME.FOR file to establish a link between the model variables used in the subroutine and the simulator's internal data structures. When a subroutine is compiled it will 'include' the EQNAME.FOR within the subroutine definition.

The file ORDER.FOR sets the major model parameters, such as length of data lags and leads, and also establishes the link between the model subroutines and the simulator. The ordering of CALLs within the file of the subroutines establishes the ordering that will be used by the Gauss-Seidel algorithm to solve the model.

The simulator uses a binary database which is built by a utility program from an ASCII data file. The binary database is built based on the list of variables that have been defined in the variables listing file. If variables are declared in the variables listing file which are not found in the ASCII data file, the system will be unable to proceed. For this reason, the names of variables used in the model should match the names that appear in the source data file.

Finally, the model residuals are built using a utility program to convert the model so that the left and right-hand sides of equations can be isolated and differences taken between the two. There are some provisions in the program to allow for special case treatment of residuals. The converted model is run and the system will use the previously generated binary database to create the residuals. The new database is then used as the input data to the simulator program created by the user.

Checklist for Model Building

(1) Write Fortran subroutines (compile to check syntax)
(2) Build variables list
(3) Build UTILITY program
(4) Create EQNAME.FOR file
(5) Convert Fortran subroutines to generate residuals
(6) Compile converted subroutines

(7) Build new UTILITY program
(8) Build binary database
(9) Build binary database with residuals
(10) Build simulator program
(11) Simulate

VI Revision of Existing Model

This section is intended as a quick guide to modifying an existing model where the variables will remain unchanged. If the model variables are to be changed, then the variables listing file will have to be modified therefore requiring virtually all the steps outlined in section V to be carried out. If, however, the changes are limited to modifying the equation form, then many of the steps in section V can be omitted and the modifications will require less time to conduct.

Changing an equation will alter the residual of the model thus requiring a modification to the database. There are two methods of implementing such a change which can be thought of as either a quick fix for experimental purposes or a permanent change to the model.

The quick fix can be accomplished by changing the equation in the subroutine, compiling the subroutine and then linking it back into the model. The residual can then be adjusted by using menu item 6 in the data Routines (itself menu item 9 in the main menu). This will require the user to manually enter the revised residual values but shortens the process so that experiments can be performed relatively quickly. Since the revisions that are entered can be saved in a start-up file this operation will only have to be performed once.

The permanent change requires that a new database with the correct residuals be created. In order to create the new database, the Fortran subroutine will have to be converted to generate the residuals. This is the step outlined in section IV.3. If only one sector has been modified then the procedure should be carried out for that sector alone. The previously built converted files will not be required to be re-compiled if the files with the .OBJ extension have not been deleted.

The next step will be to generate the residuals from the converted files. This step is outlined in section IV.4. Since the data that the converted programs require are already in the a binary database, we can specify file 7 as the binary database which had been already been built for simulations (file 7 in the

simulator files listing menu). Following the remaining steps in section IV.4 will create a new database for the revised model so that the only step remaining will be to link the object file with the revised equations to the rest of the object file to create a new simulator program.

The following is an example of this procedure that uses the INTERMOD model to demonstrate the steps required to change an equation. The INTERMOD model is contained in the 10 files: G7CAN.FOR, G7USA.FOR, G7JAP.FOR, G7GER.FOR, G7FRA.FOR, G7ITA.FOR, G7UKM.FOR, G7SI.FOR, G7ROW.FOR and G7DC.FOR. The converted residual object files will be contained in files which are terminated by RES before the extension, eg. G7CANRES.OBJ. If an equation in the Canadian sector - G7CAN.FOR - is to be modified then a new converted file will have to be built. The program to create this new file is itself built with the command[4]:

```
LINK UTILITY+ORDER+SCREEN+(SUBS) /E[5]
```

The UTILITY program that this command builds is then run using the start-up file DECK.UTL. Following the instructions of section IV.3 and specifying G7CANRES.FOR as the converted model file will create a program which must then be compiled. Issuing the command:

```
FL /AH /c G7CANRES.FOR
```

will create a file called G7CANRES.OBJ. This file is then linked to the UTILITY program with the command:

```
LINK UTILITY+ORDER+SCREEN+(SUBS)+(G7CANRES+G7USARES+
     G7JAPRES)+(G7GERRES+G7FRARES+G7ITARES)+(G7UKMRES+
     G7SIRES+G7ROWRES+G7DCRES) /E[6]
```

---

[4]This step can be omitted if a version of UTILITY already exists.

[5]Unless the ORDER.FOR program is modified to remove the calls to the subroutines, this command will produce error messages from the LINK program. These messages can be ignored.

[6]For the users who are more familiar with the DOS operating system, the files LINK and LINKRES contain lists of the modules that are to be linked together. The command:

```
LINK @LINKRES
```

will accomplish the same task as the command line given above. These files can be modified for use with other models.

The user should press <ENTER> in response to the questions asked by the LINK program.

The instructions given in section IV.4 will then create a new binary database and a new model can be built to use this database by issuing the command:

```
LINK INTER1+INTER2+ORDER+SCREEN+G7CAN+G7USA+G7JAP+G7GER+
     G7FRA+G7ITA+G7UKM+G7SI+G7ROW+G7DC /E
```

By responding SIM to the first prompt the LINK program issues and pressing <ENTER> for subsequent prompts, the program SIM will be created.

VII Miscellaneous Tips

(1) Having created the residuals for a model we can set the newly created database as the input database and rerun the model through the UTILITY program, without invoking the option to create residuals. Since all equations will contain their calculated residuals there should be no difference between the database values of endogenous variables and the values calculated by the model. If any of the identities in the model are not calculated properly, or if residuals have not been entered properly, the program will detect a difference between predicted values and observed values and will signal the difference.

The maximum number of iterations should be set to 1 in this procedure to avoid unnecessary calculations.

(2) When running the model, setting the maximum number of iterations to a low value and then increasing it in subsequent simulations will allow the evolution of variables in the model to be monitored through the data routines. The technique is useful when a model is 'exploding' since it allows the user to precisely determine which variables are causing the model to become unstable.

(3) By setting the maximum number of iterations to -1 and then requesting the system to save a control solution, the historical data will be put into the control solution file. This will allow shock minus control analysis to examine the differences between history and a shock solution.

---

[7]Assuming that the file G7CAN.FOR was re-compiled to create G7CAN.OBJ.

ERRATA

Please note that the attached appendices were not included in:

Department of Finance
Working Group on International Macroeconomics
Working Paper #88-9

INTERMOD Model User's and Developer's Guide
for PC Systems by P. Bagnoli and A. Kruus

Please add them to your copy of the above paper.

Appendix I

```
&ENDOGENOUS
UA       UB       UC       UCU      UNFA     UGDP     UGNP     UK       UM       UP
UPGNP    UPIE     UPIM     URL      URLE     URS      UTAX     UTRATE   UUCST    UXT
UYCAP    UYD      UIM      UCRBL    UFM      UGDEF    UGE      UPI      UPIMA    UR
UREER    URLR     URSR     UXM      UINV     XXX1     UCOIL    UPXM     UXMA     UIOIL
UICOM    UPXT     UIT      UPIT     UER      UW       UPFM     UMERM    UNNP     UNNPC
URR      UPNO
&EXOGENOUS
UG       UBRAT    UDLTA    UGXOG    UMULT    UPIBR    UCTRT    UXOIL    UPRDL    UTXCD
UMT
&RESIDUALS
RUA      RUB      RUC      RUNFA    RUGDP    RUGNP    RUK      RUP      RUPGN    RUPIE
RUPIM    RURL     RURLE    RURS     RUTAX    RUIVT    RUYCP    RUCOL    RUTRT    RUIM
RUXM     RUPXM    RUOIL    RUICM    RUIT     RUXT     RUM      RUW
&COEFFICIENTS
 KIC0          .687478 KIC1          2.401260 KIC2          -.515604
 KIC3         -.179167 S95            .065300 FW2            .391139
 KIC4          .151221 KIC5          -.594397 JPXM3          .000000
 KPXM0         .243312 W57            .048900 GC1            .116817
 HI3           .368385 SC5            .073380 W41            .182800
 S56           .070100 CC0           -.395676 SX3           -.281500
 V99           .000000 FM0          -1.017480 V83            .000000
 W18           .242800 X64            .125200 UP1            .146192
 S17           .770600 KRHBR         1.000000 FC4            .000000

12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

Appendix II

```
C
      SUBROUTINE SOLV11(NTIME,YZ)
C
$INCLUDE: 'EQNAME.FOR'
C
C   Expected future long term interest rate :
C
      URLE = FORWRD*LEAD(1,URL)+
     1       (1-FORWRD)*(LAG(1,URLE)+ETAR*(URL-LAG(1,URLE))+RURLE)
C
C
C  Real ex ante short rate
C
      URSR = (1+URS/100)/(1+URLE)-1
C
C  Net national product
C
      UNNP = UGDP-UDLTA*LAG(1,UK)

C
C  capacity net national product
C
      UNNPC = UYCAP-UDLTA*LAG(1,UK)+RUNNPC
C
C   Wealth :
C
      UW = ULMBDM*UM/UP+0.5*(1+((LAG(3,URL)+LAG(2,URL)+LAG(1,URL))

     1      /3)/URL)*UNFA/UER/UP+UNNPC*(1-UGBAR)*URHBR*(1/URRBR+
     2      (UCU*0.01-1)/(URRBR+UW1)+(UPGNP/UP/URHBR-1)/(URRBR+UW2)
     3      -(URSR-URRBR)/(URRBR+UW3)-((UG*UP+UGXOG)/(UNNP*UPGNP)
     4      -UGBAR)/(1-UGBAR)/(URRBR+UW4))+RUW
C
C   Average tax rate (reaction function) :
C
      UTRATE = LAG(1,UTRATE)+TAU1*DUM*(LAG(1,UB)/LAG(1,UPGNP)
     1      /LAG(1,UGNP)-UBRAT)+TAU2*DUM*((LAG(1,UB)/LAG(1,UPGNP)
     2      /LAG(1,UGNP))-(LAG(2,UB)/LAG(2,UPGNP)/LAG(2,UGNP)))+
     3      RUTRT
C
C   Consumption :
C
      UC = EXP(LOG(LAG(1,UC))+UC0+UC1*LOG(LAG(1,UW)/LAG(1,UC))+
     1     UC2*(URLR-LAG(1,URLR))+UC3*(LOG(UYD/LAG(1,UYD)))+RUC)
C
      RETURN
C
      END
```

```
C **** ORDER ***********************************************************
      SUBROUTINE ORDER(DUM,NTIME,Y)
C
      LOGICAL DUM(30)
      REAL Y(1)
      INTEGER NTIME
      IF (DUM(20)) CALL SOLV20(NTIME,Y)
      IF (DUM(19)) CALL SOLV19(NTIME,Y)
      IF (DUM(11)) CALL SOLV12(NTIME,Y)
      IF (DUM(11)) CALL SOLV11(NTIME,Y)
      IF (DUM(12)) CALL SOLV21(NTIME,Y)
      IF (DUM(12)) CALL SOLV22(NTIME,Y)
      IF (DUM(13)) CALL SOLV31(NTIME,Y)
      IF (DUM(13)) CALL SOLV32(NTIME,Y)
      IF (DUM(14)) CALL SOLV41(NTIME,Y)
      IF (DUM(14)) CALL SOLV42(NTIME,Y)
      IF (DUM(15)) CALL SOLV51(NTIME,Y)
      IF (DUM(15)) CALL SOLV52(NTIME,Y)
      IF (DUM(16)) CALL SOLV61(NTIME,Y)
      IF (DUM(16)) CALL SOLV62(NTIME,Y)
      IF (DUM(17)) CALL SOLV71(NTIME,Y)
      IF (DUM(17)) CALL SOLV72(NTIME,Y)
      IF (DUM(18)) CALL SOLV81(NTIME,Y)
      IF (DUM(18)) CALL SOLV82(NTIME,Y)
      RETURN
      END
C
C ***** SETMOD ***********************************************************
      SUBROUTINE SETMOD(SECTRS,NL,NF)
C
      CHARACTER*6 SECTRS(1)
      INTEGER NL,NF
C
C This subroutine sets up the sectors that the model will use.  A total of
C 30 sectors can be defined which are assigned as SECTRS(1)...SECTRS(30).
C The sectors that are set up here must be called from the ORDER subroutine.
C
      SECTRS(11) ='USA'
      SECTRS(12) ='JAPAN'
      SECTRS(17) ='CANADA'
      SECTRS(13) ='GERMAN'
      SECTRS(14) ='FRANCE'
      SECTRS(16) ='ITALY'
      SECTRS(15) ='UKM'
      SECTRS(18) ='SMALL'
      SECTRS(19) ='LDC'
      SECTRS(20) ='ROW'
C
C  Set number of lags used in model. Maximum is 5
C
      NL=3
C
C  Set number of leads used in model. Maximum is 5
C
      NF=4
C
      RETURN
      END
```

Appendix IV

UA
UB
&UC
UNFA
UGDP
UGNP
&UK
%UP
&UPNO
UPIE
UPIM
%URL
URLE
URS
UTAX
UINV
&UYCAP
&UCOIL
%UTRATE
&UIM
&UXM
&UPXM
UIOIL
&UICOM
UIT
UXT
%UM
UW